OpenGL Techniques for Snow Leopard

Session 310

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

OpenGL Techniques for Snow Leopard

Chris Niederauer
Senior GPU Software Engineer

What Is OpenGL?

- API to interact with a GPU
- Provide a 3D cinematic experience
- Advanced user interface effects

Window Effects

Image Effects

Video Effects

OpenGL Programming Resources

- OpenGL Programming Guide (7th Edition)—"The Red Book"
- OpenGL Shading Language (3rd Edition)—"The Orange Book"
- Apple OpenGL ADC site:
  http://developer.apple.com/opengl/

OpenGL in Snow Leopard

Overview

- Dealing with multiple GPUs
- Writing well behaved fullscreen applications
- Multithreading and multiple contexts
- Sharing between OpenCL and OpenGL

OpenGL in Snow Leopard

New OpenGL extension support

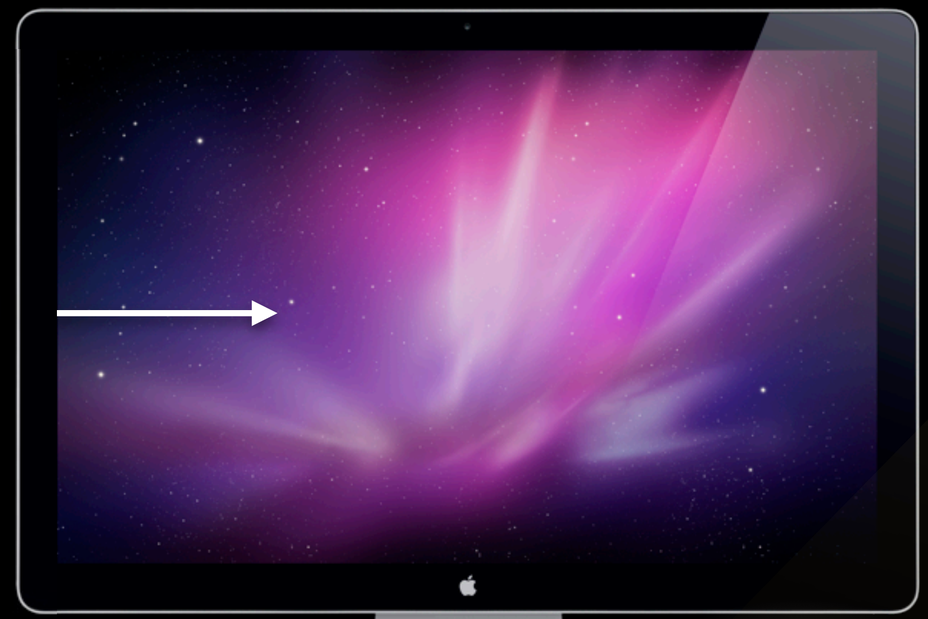
- APPLE_rgb_422
- ARB_color_buffer_float
- ARB_half_float_vertex
- ARB_texture_rg
- ARB_texture_compression_rgtc
- EXT_framebuffer_sRGB
- EXT_framebuffer_blit *(also available in Leopard)*
- EXT_framebuffer_multisample *(also available in Leopard)*

Managing Multiple GPUs

- Many Macs support multiple GPUs
  - Built-to-order MacPro with four GPUs
- Increasingly important
- Apps need to support dynamic environments, not just your MacBook!

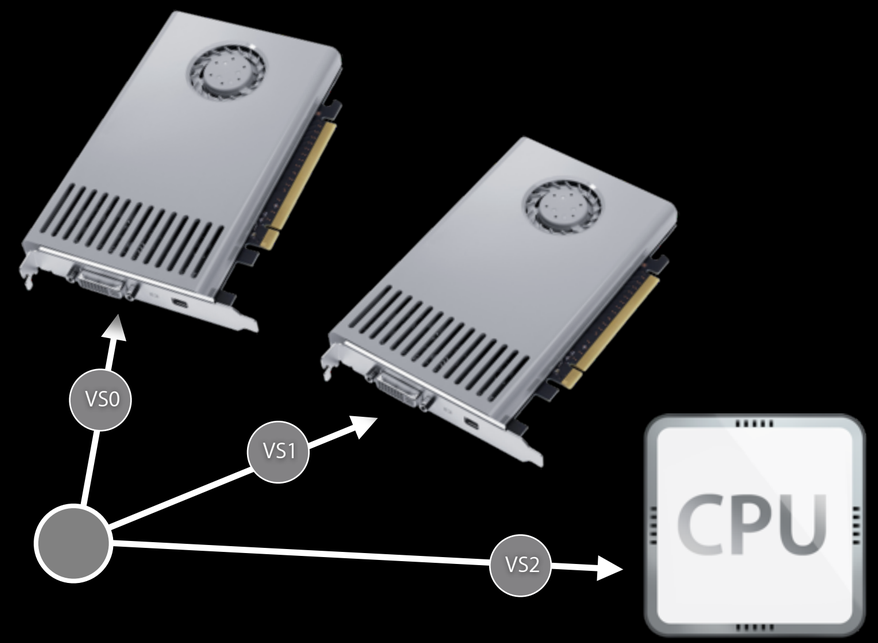
Multi-Monitor Multi-GPU Example

Seamlessly moving between GPUs



Virtual Screens

- Each hardware device is a virtual screen
- The software renderer is a virtual screen
- Unrelated to physical displays



- Virtual screen list is chosen by `CGLChoosePixelFormat`
- Context virtual screen is set by
  - `CGLUpdateContext` / `[ctx update]` (implicit update)
  - `CGLSetVirtualScreen` / `[ctx setCurrentVirtualScreen:]`
  - `CGLSetPBuffer`

Multiple Displays with Unique GPUs

Tracking when a view switches GPUs

- Maintain the correct virtual screen
  - Initialize virtual screen at startup
  - Check at all CoreGraphics display change notifications if a context has moved and ensure the capabilities are valid
- Surfaces unrelated to a display need to be manually managed

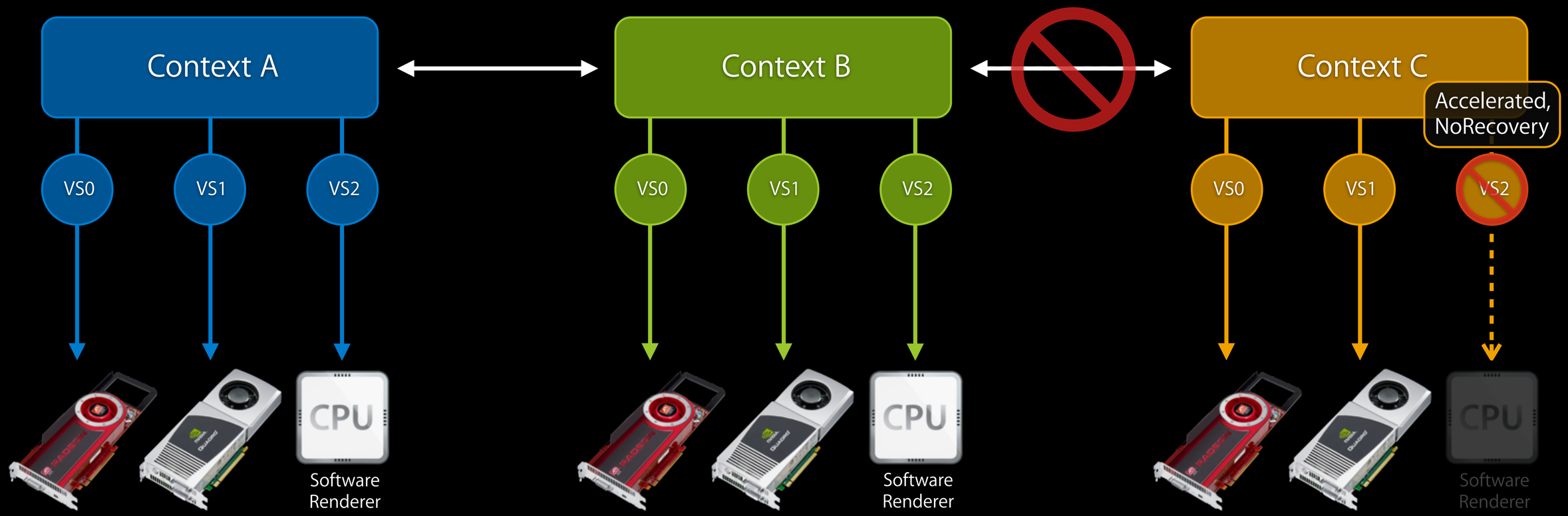
The Code…

```
- (void) update {
    [super update];
    newVirtualScreen = [self currentVirtualScreen];
    if(newVirtualScreen != oldVS) {
        // Check for GL support
        // Move any related contexts over
        oldVS = newVirtualScreen;
    }
}
```

- PBuffers and front/back buffers are not preserved on switch
- Not subclassing NSOpenGLView?
  - Register for CoreGraphics display change notifications

Sharing Contexts

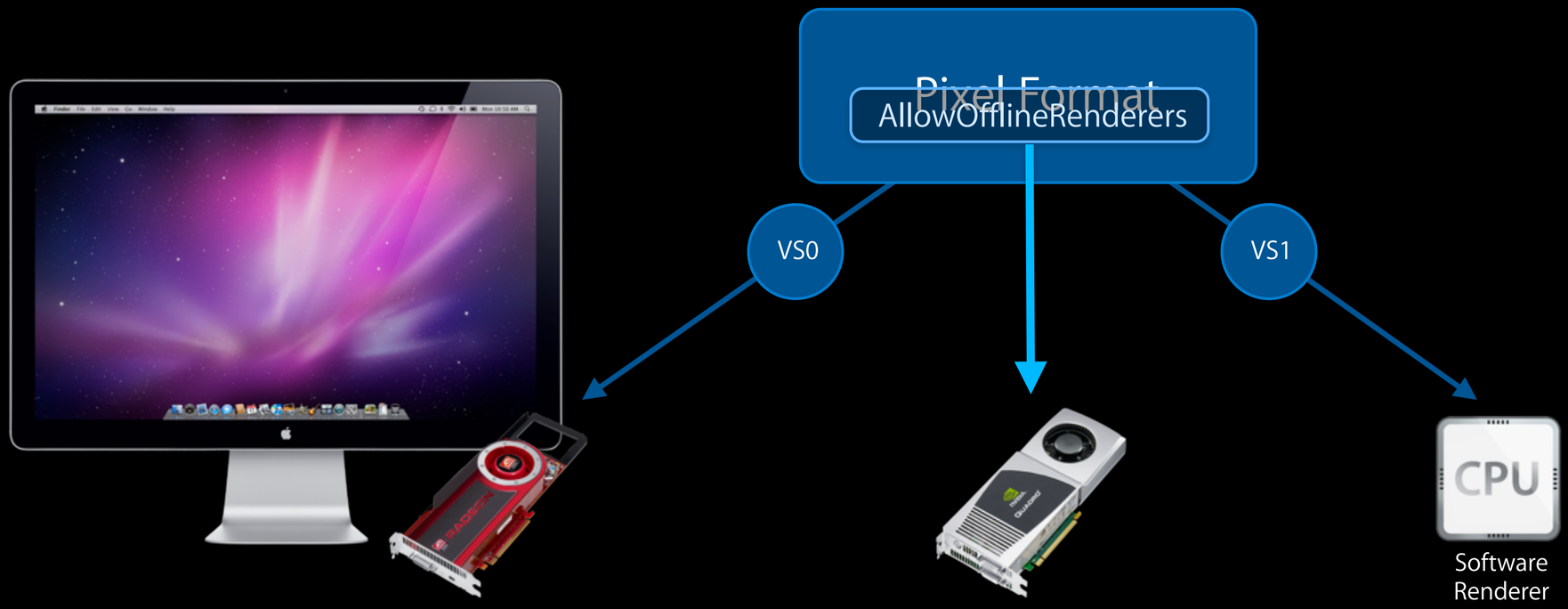
Maintain similar virtual screens



Offline Renderers

And surviving a hot-plug

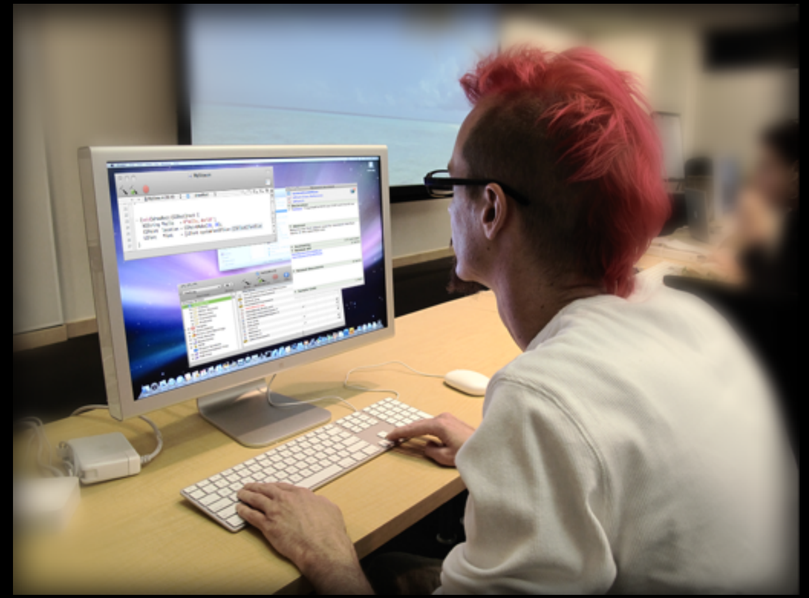
- You may have another GPU without a display on
- You may want another GPU to come online at a later date
- Add NSOpenGLPFAAllowOfflineRenderers to the pixel format attribute list



Our Recommendation

Test on a system with two GPUs and two displays

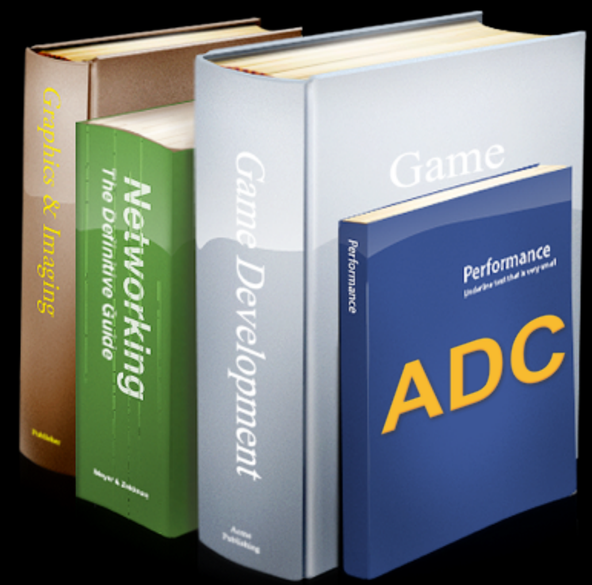
- Test that OpenGL rendering occurs on the same GPU that displays it
- Test hot-plugging a single display into multiple GPUs



More Multi-GPU Information

- ADC Tech Note (TN 2229)

  Supporting Multiple GPUs on Mac OS X



Full Screen Rendering

Why is it deprecated?

- Does not interact well with multiple GPUs
  - Can not always share contexts
  - Contexts must be created from scratch
- Covers up critical dialogs

There's a Better Way to Do It

It just works!

- Covering contexts automatically get benefits of "fullscreen"
  - Cache enhancements
  - Page flipping

Key Code Points

- Make a screen sized window

```
NSRect mainDisplayRect = [[NSScreen mainScreen] frame];
NSWindow win = [[NSWindow alloc] initWithContentRect:mainDisplayRect
                                                styleMask:...];
[win setContentView:myFullScreenNSOpenGLView];
```

- Set the window level above the Menu Bar

```
[myWindow setLevel:NSMainMenuWindowLevel+1];
```

FullScreenNSOpenGLView Example

Chris Niederauer
GPU Ninja

Deprecated API

- `kCGLPFAFullScreen`
- `CGLSetFullScreen()`
- Can’t depend on back buffer content!
  - `kCGLPFABackingStore` will prevent the performance gain

OpenGL and Multithreading

Avoiding land mines

- OpenGL is not per context thread-safe
- Convenience functions
  - CGLLockContext(ctx) / CGLUnlockContext(ctx)
  - CGLEnable(kCGLCEMPEngine)
- Or multithread everything else!

- ADC documentation

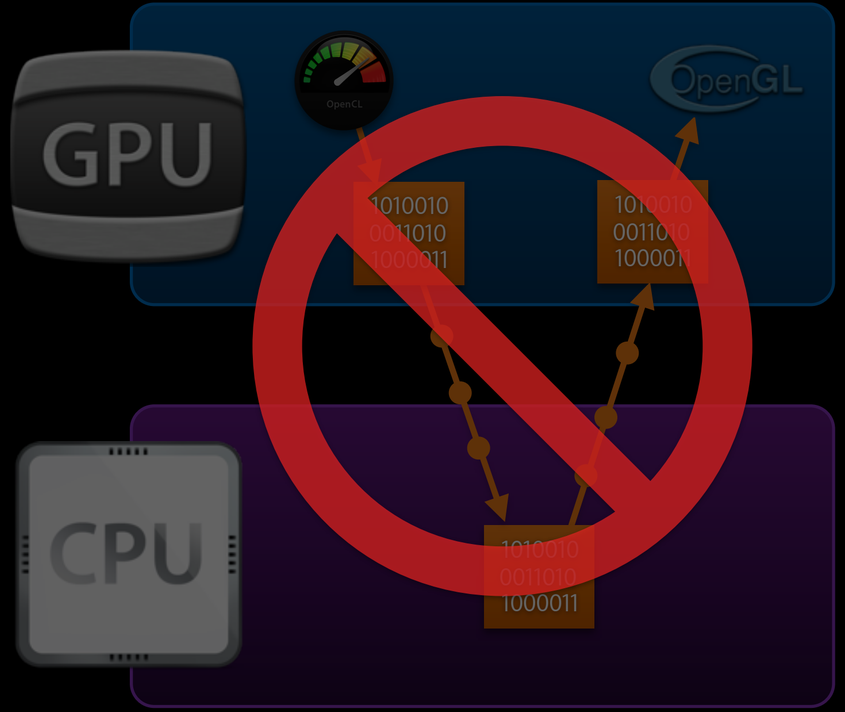
  OpenGL Programming Guide for Mac OS X: Multithreading and OpenGL

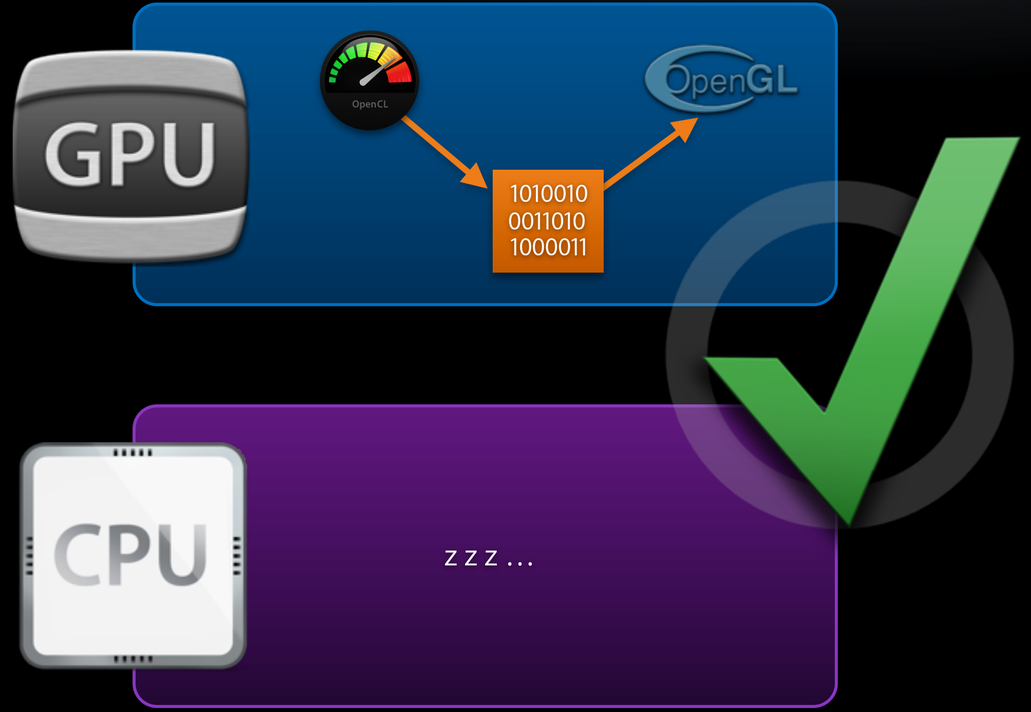
OpenCL

Working with OpenCL

The GPU goes to 11!

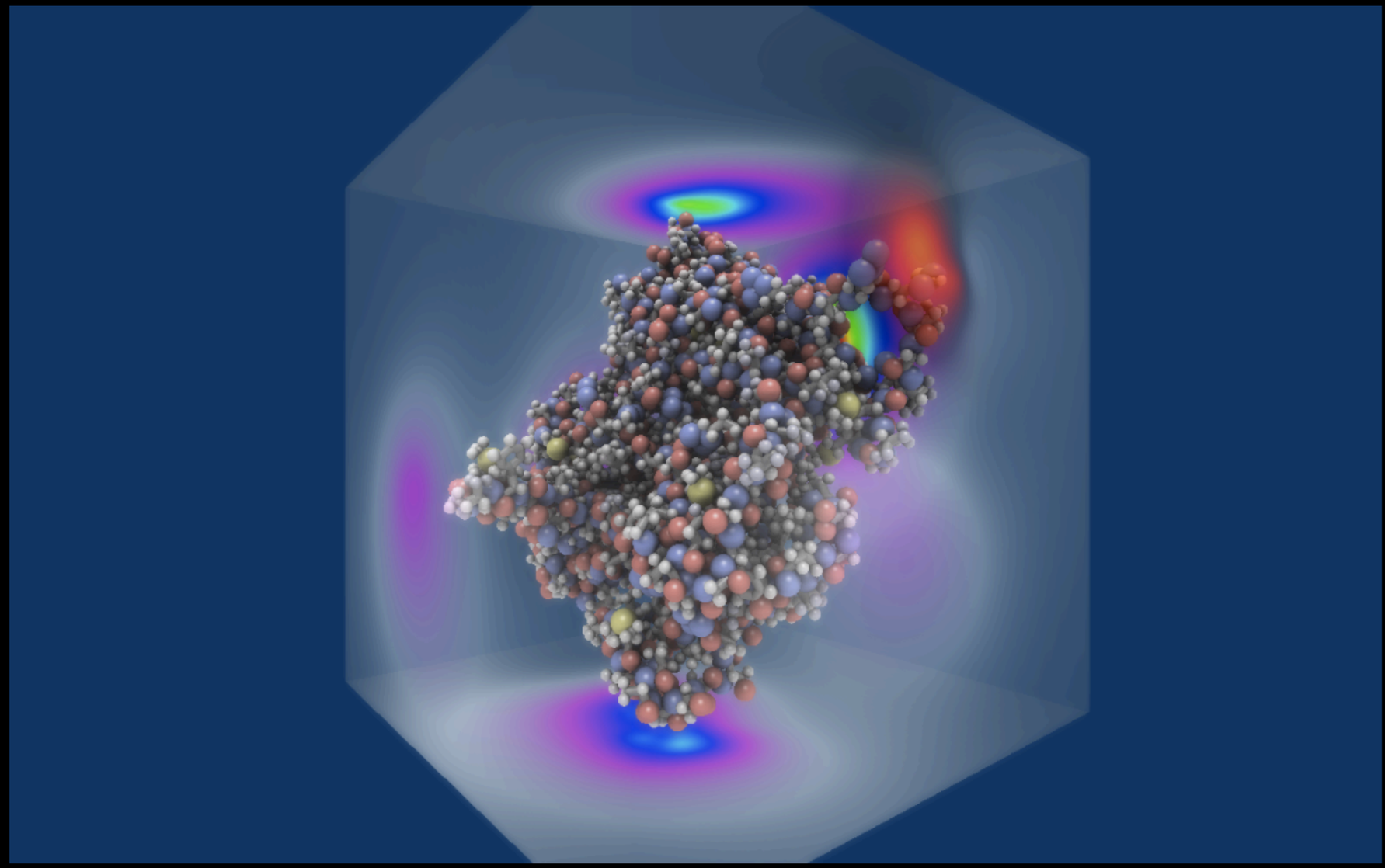
- OpenCL creates data on the GPU
- OpenGL consumes data on the GPU





OpenCL Sharing Texture Data

Molecule



Working with OpenCL

Basic OpenGL/OpenCL interactions

- Creating a CL context from a GL context
- Creating CL buffers from GL buffer objects
- Creating CL images from GL textures
- Sharing buffers and textures
- Supporting multi-GPU

Sharing Between OpenCL and OpenGL

Creating a CL context from a GL context

```
#include <OpenCL/cl_ext.h>
#include <OpenCL/cl_gl.h>
```

```
// Test for cl_APPLE_gl_sharing extension support and then find the
// clGetGLContextInfoAPPLE function pointer via clGetExtensionFunctionAddress
```

```
// Create a context that shares with an OpenGL context
CGLShareGroupObj share_group = CGLGetShareGroup(cgl_ctx);
cl_context_properties properties = { CL_CONTEXT_PROPERTY_USE_CGL_SHAREGROUP_APPLE,
                                     (cl_context_properties)share_group, 0 };
cl_context cl_ctx = clCreateContext(properties, 0, 0, 0, 0, &err);
```

```
// Compute OpenCL data on the same GPU as the OpenGL context
cl_device_id currentDevice;
err = clGetGLContextInfoAPPLE(cl_ctx, cgl_ctx,
          CL_CGL_DEVICE_FOR_CURRENT_VIRTUAL_SCREEN_APPLE,
          sizeof(cl_device_id), &currentDevice, NULL);
```

Sharing Between OpenCL and OpenGL

Creating CL buffers from GL buffer objects

```
glGenBuffers(1, &vboName);
glBindBuffer(vboName);
glBufferData( … );

cl_mem buffer = clCreateFromGLBuffer(cl_ctx, CL_MEM_READ_WRITE, vboName, &err);
```

Sharing Between OpenCL and OpenGL

Creating CL images from GL textures

```
glGenTextures(1, &texName);
glBindTexture(GL_TEXTURE_2D, texName);
glTexImage2D( … );

cl_mem buffer = clCreateFromGLTexture2D(cl_ctx, CL_MEM_READ_ONLY,
                                        GL_TEXTURE_2D, 0, texName, &err);
```

- Similar mechanism for other types:
  - `clCreateFromGLTexture3D( … )`
  - `clCreateFromGLRenderbuffer( … )`

Sharing Between OpenCL and OpenGL

Sharing buffers and textures

```
glFlush(); // use glFlushRenderAPPLE() on single buffered contexts

// Start sharing an OpenCL buffer
err = clEnqueueAcquireGLObjects(queue, 1, &buffer, 0, NULL, NULL);

// Do some CL operations like executing a kernel
// ...

err = clEnqueueReleaseGLObjects(queue, 1, &buffer, 0, NULL, NULL);
```

Sharing Between OpenCL and OpenGL

Supporting multiple GPUs

```
cl_device_id  deviceForVirtualScreen[32];
size_t        param_value_size_return;
clGetGLContextInfoAPPLE(cl_ctx, cgl_ctx,
    CL_CGL_DEVICES_FOR_SUPPORTED_VIRTUAL_SCREENS_APPLE,
    32 * sizeof(cl_device_id),
    &deviceForVirtualScreen[0], &param_value_size_return);

int numDevices = param_value_size_return / sizeof(cl_device_id);
```

```
- (void) update { // NSOpenGLView subclass method
    [super update];
    curVirtualScreen = CGLGetVirtualScreen(cgl_ctx);
    if (curVirtualScreen != previousVirtualScreen) {
        // Ensure supported OpenGL context movement
        curDeviceID = deviceForVirtualScreen[curVirtualScreen];
    }
}
```

The Tools

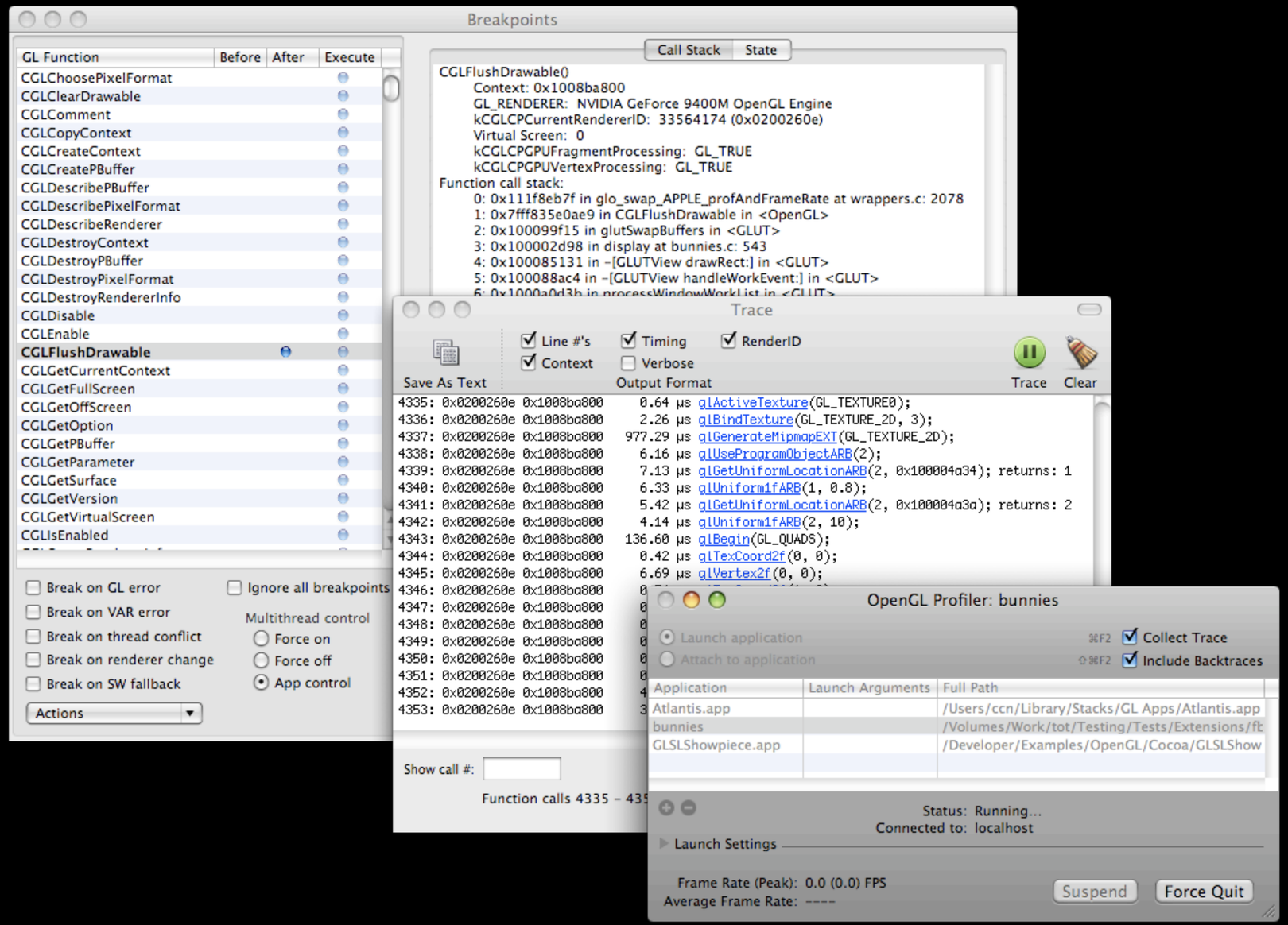
Critical to analyze performance

- OpenGL Profiler
- OpenGL Driver Monitor
- OpenGL Shader Builder
- Shark and Instruments

OpenGL Profiler

Performance and correctness debugger

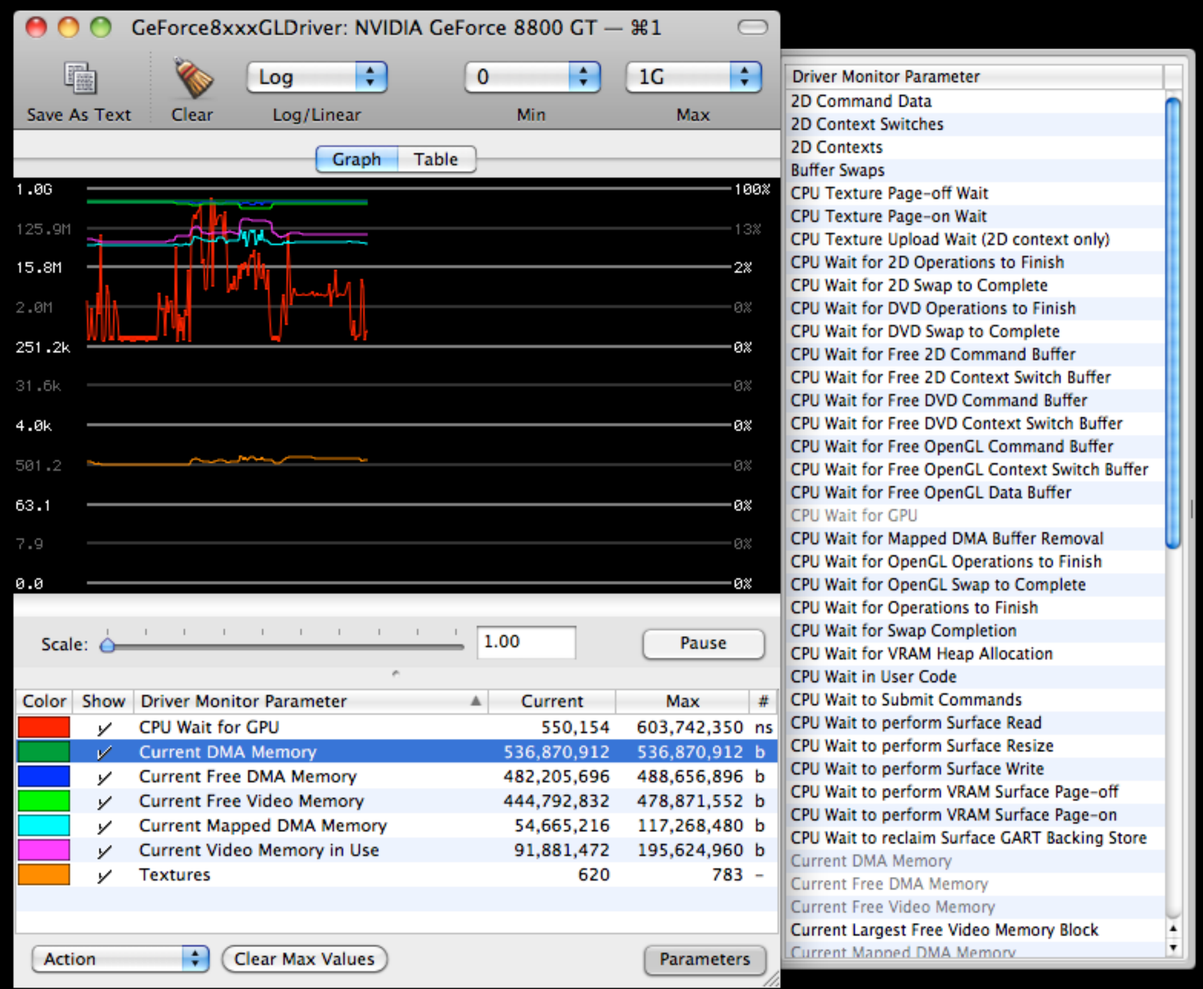
- Set breakpoints
  - GL entrypoints
  - GL errors
  - Thread conflicts
  - Renderer changes
  - Software fallback
- Trace GL calls
- View GL state
- View GL objects
- View the draw buffer



OpenGL Driver Monitor

Driver counters and statistics

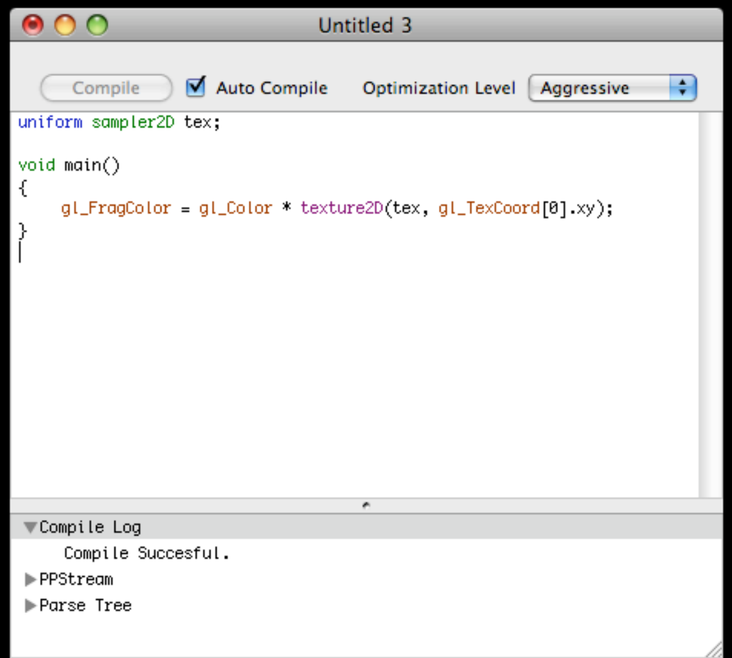
- Monitor CPU stalls
- Find leaks and paging
- Interesting driver statistics
  - CPU wait for GPU
  - Current free video memory
  - Surface page off data
  - Texture page off data

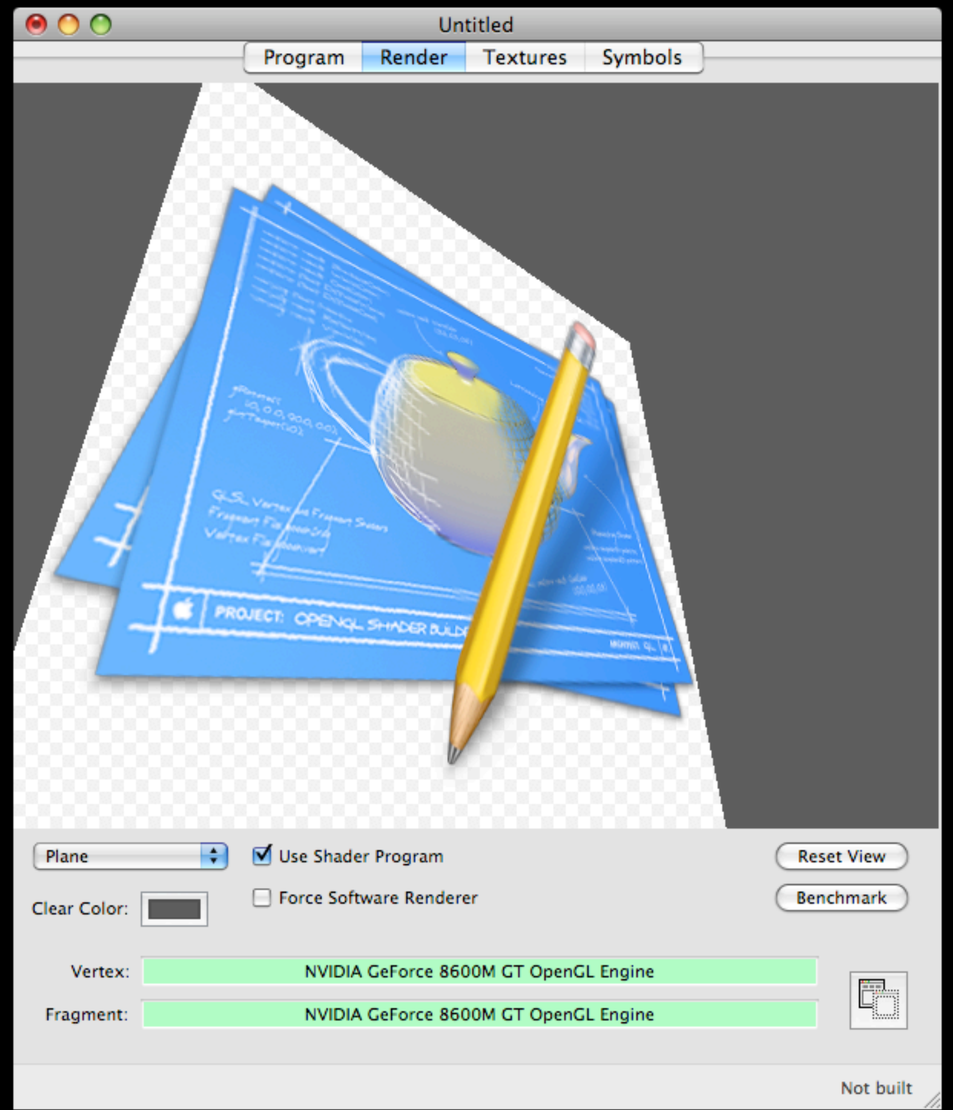


OpenGL Shader Builder

Vertex, geometry and fragment program editor

- GLSL and ARB assembly languages
- Benchmark shader performance

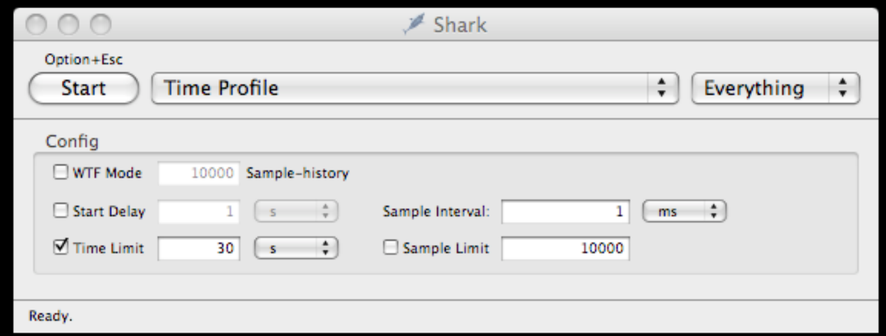




Shark

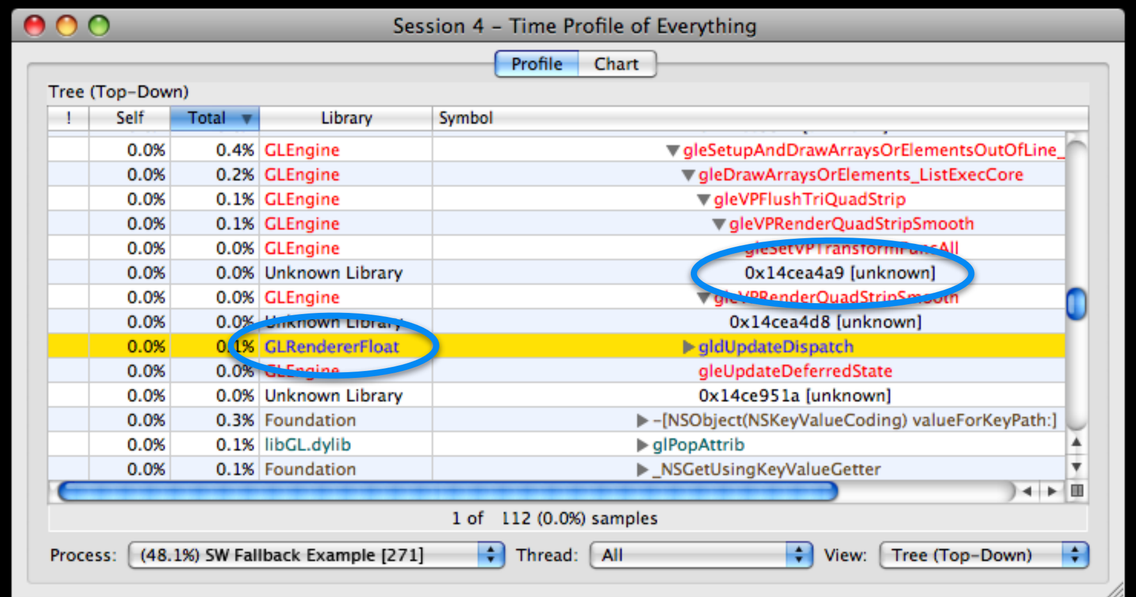
Code profiler

- Look out for these symbols!
  - Software Renderer: GLRendererFloat
  - RTC symbols: 0x4DD4355 [unknown]
  - MT Sync: gleFinishCommandBuffer



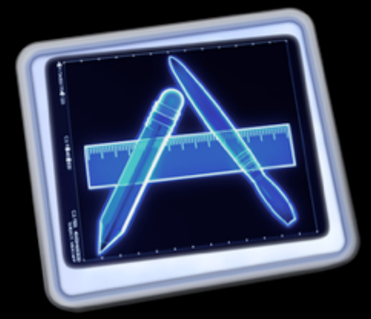
- Drivers symbols are ambiguous
  - gldGetTextureLevel+0x1234

- New to Shark? Go see the sessions!



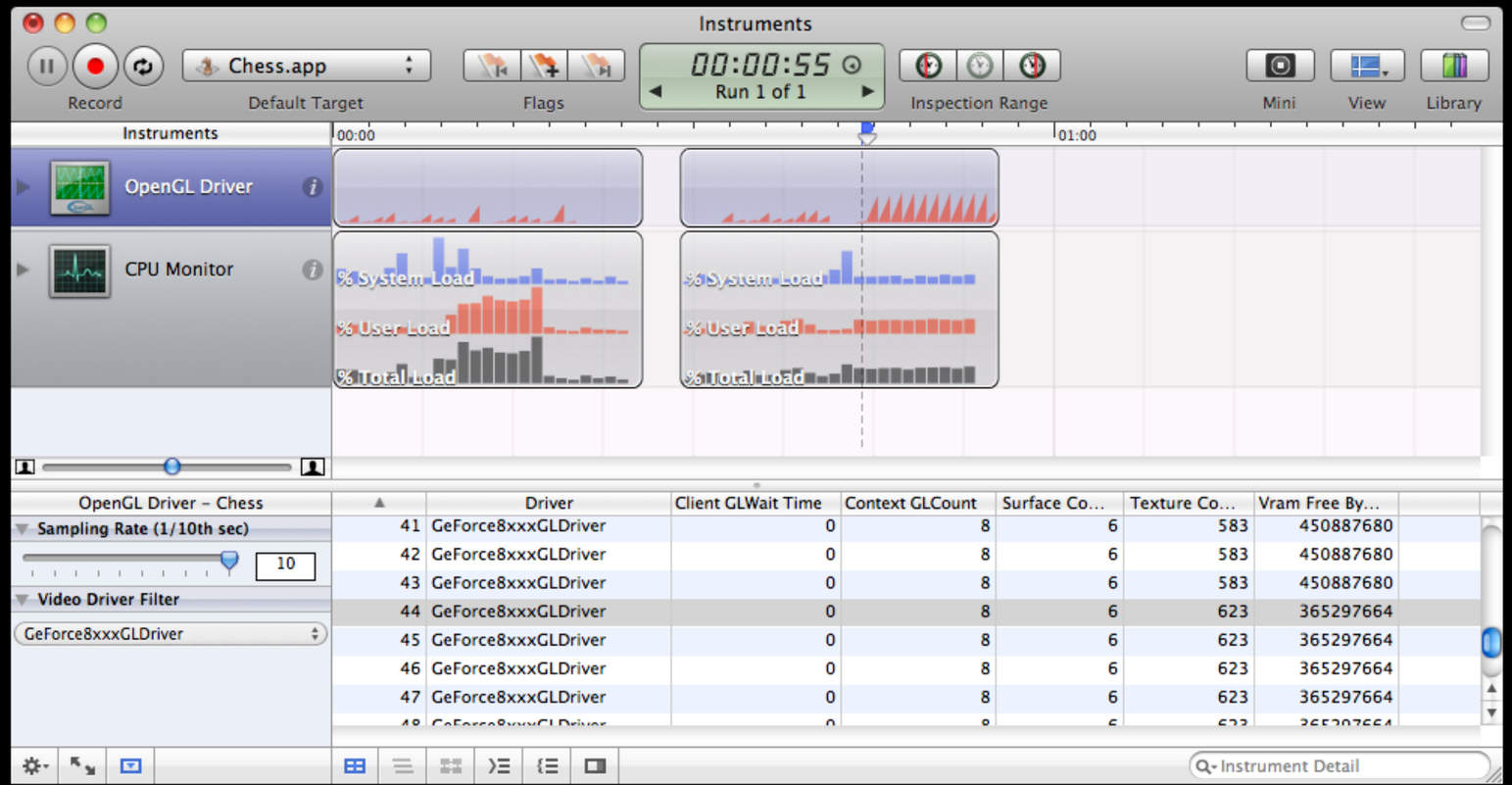
Instruments

Visualize application performance

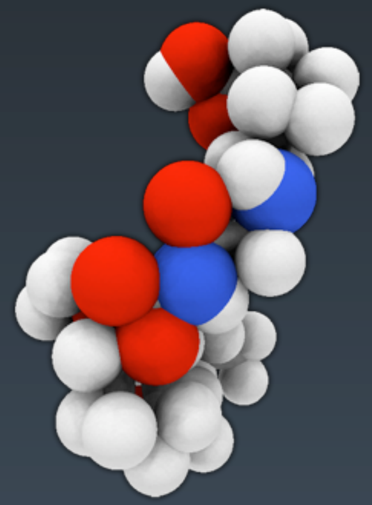


- With integrated OpenGL driver monitor statistics

  Remember! GPU statistics are per GPU, not per application



Molecules Demo



Chris Niederauer
Once beat Chuck Norris in Quake

More Information

Allan Schaffer
Graphics Technology Evangelist
aschaffer@apple.com

Documentation

Mac OS X OpenGL Developer Site
http://developer.apple.com/opengl/

Related Sessions

OpenGL ES Overview for iPhone OS	OpenGL ES	Presidio Wednesday 2:00PM
Advanced Rendering and Shading	OpenGL OpenGL ES	Presidio Wednesday 3:30PM
OpenGL ES Tuning and Techniques	OpenGL OpenGL ES	Presidio Wednesday 5:00PM
Introduction to OpenCL	OpenCL	Marina Thursday 9:00AM
Harnessing the Power of OpenCL	OpenCL	Marina Thursday 10:30AM
Mastering OpenCL	OpenCL	Marina Thursday 2:00PM
Performance Tuning with Shark on Mac and iPhone		North Beach Friday 9:00AM

Labs

OpenGL for the Mac Lab	Graphics and Media Lab A Thursday 12:00PM
OpenGL ES Lab	Graphics and Media Lab B Thursday 12:00PM
OpenCL Lab	Graphics and Media Lab B Thursday 3:30PM

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09